Reconstructing syntax in RefactorErl

Kitlei Róbert

Eötvös Loránd University, Budapest, Hungary
CEFP 2009, Budapest-Komárom

May 27, 2009

Overview

RefactorErl

- a refactoring tool for Erlang

RefactorErl

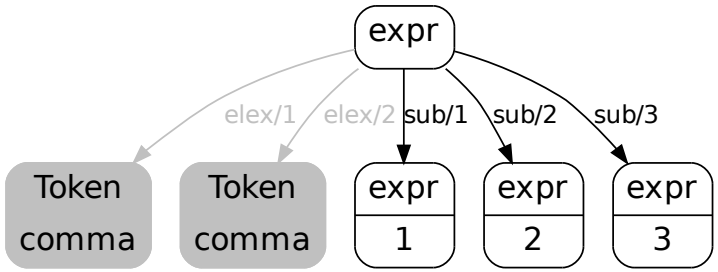
- a refactoring tool for Erlang
- graph representation
  - based on the syntax tree
  - analyser modules add static semantic information
  - node contractions and labelled edges

RefactorErl

- a refactoring tool for Erlang
- graph representation
  - based on the syntax tree
  - analyser modules add static semantic information
  - node contractions and labelled edges
- the graph is stored in a database
  - fast, simple, composable queries
  - the order of the edges is not fully preserved

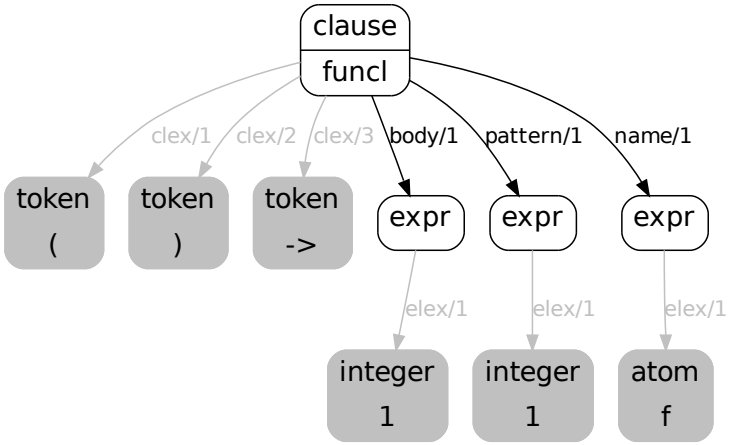
Node contractions

- nodes are contracted into groups in two ways
  - chain rules are eliminated
- repetitions are grouped under one node
  - 3 groups for Erlang: form, clause, expr



Labelled edges

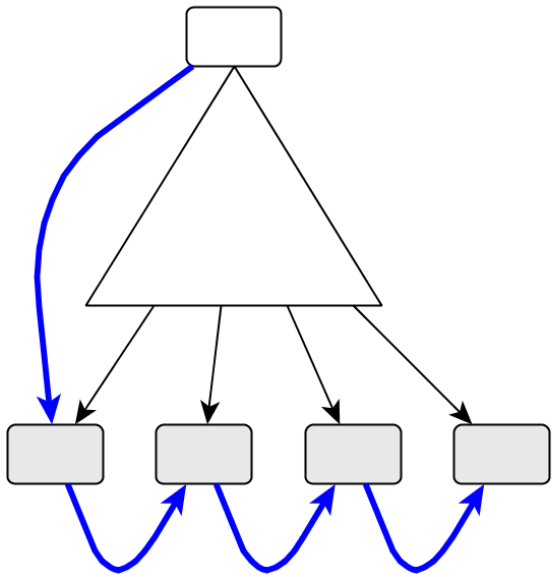
- edges are labelled according to their use
  - example: the graph representation of `f(1) -> 1.`



- order is preserved only within edges of the same label
  - cause: the database has one table per edge label
- for reprinting the file, the full order has to be restored

Restoring the order

- one possibility: link the tokens with a special edge
  - this solution was used in the previous versions of RefactorErl
  - solves the ordering easily
  - extremely hard to update



Restoring the order

- another possibility: store the order in the node
  - costs extra space
  - update difficutly is similar to the solution to be presented

Restoring the order

- current solution: use the grammar description
- it has the structural information that is missing because of the partiality
  - slight restrictions on the grammar description
  - can still express everything
- reacquiring the original text
  - reorder the children of one node at a time
  - walk the tree while reordering, and collect the front of the tree